HPI Hasso Plattner Institut
IT Systems Engineering
Universität Potsdam

Master Thesis

Language Identification using Deep Convolutional Recurrent Neural Networks

<Deutscher Titel bei englischer Arbeit>

Tom Herold

tom.herold@student.hpi.uni-potsdam.de

XX.02.2016

Supverisors
Prof. Dr. Christoph Meinel
Dr. Haojin Yang

Internet-Technologies and Systems
Hasso Plattner Institute
University of Potsdam, Germany

Abstract

Zusammenfassung

Acknowledgments

I would like to express my gratitude to my supervisors Prof. Dr. Christoph Meinel and Dr. Haojin Yang. I want to thank them for giving me the opportunity to research this interesting topic, as well as for their guidance and advice. I especially want to thank Dr. Yang for his insights, support, and inspiration regarding the deep learning techniques used in my masters thesis. I would also like to thank my colleagues Georg Wiese, Christian Bartz, Tom Bocklich, Norman Rzepka, and Johannes Jasper for the many fruitful discussions about language identification and machine learning in general. I owe them a great deal of gratitude for supporting and encouraging me while working on this thesis.

Thank you.

Contents

Abbreviations

CNN convolutional neural network

GPU graphics processing unit

LID language identification

LSTM long short-term memory

NIST National Institute of Standards and Technology

SGD stochastic gradient descent

t-SNE t-distributed stochastic neighbor embedding

1. Introduction

1.1. Language Identification as the key to speech tasks

1.2. Contribution

1.3. Outline of the Thesis

2. The Language Identification Problem

2.1. Language Identification

2.2. Task Specification in This Thesis

3. Theoretical Background

3.1. Machine Learning

3.1.1. Types of Machine Learning

3.1.2. Classification

3.2. Building Blocks of Neural Networks

3.2.1. Fully Connected Layer

3.2.2. Convolutional Layers

3.2.3. Pooling Layers

3.2.4. Batch Normalization Layers?

3.2.5. Softmax Loss Function

3.3. Recurrent Neural Networks

3.3.1. Long Short Term Memory Networks

3.4. Hybrid Networks

- Convolutional Recurrent Neural Networks

3.5. Audio Representations

- MFCC
- Spectrogram
- Waveform

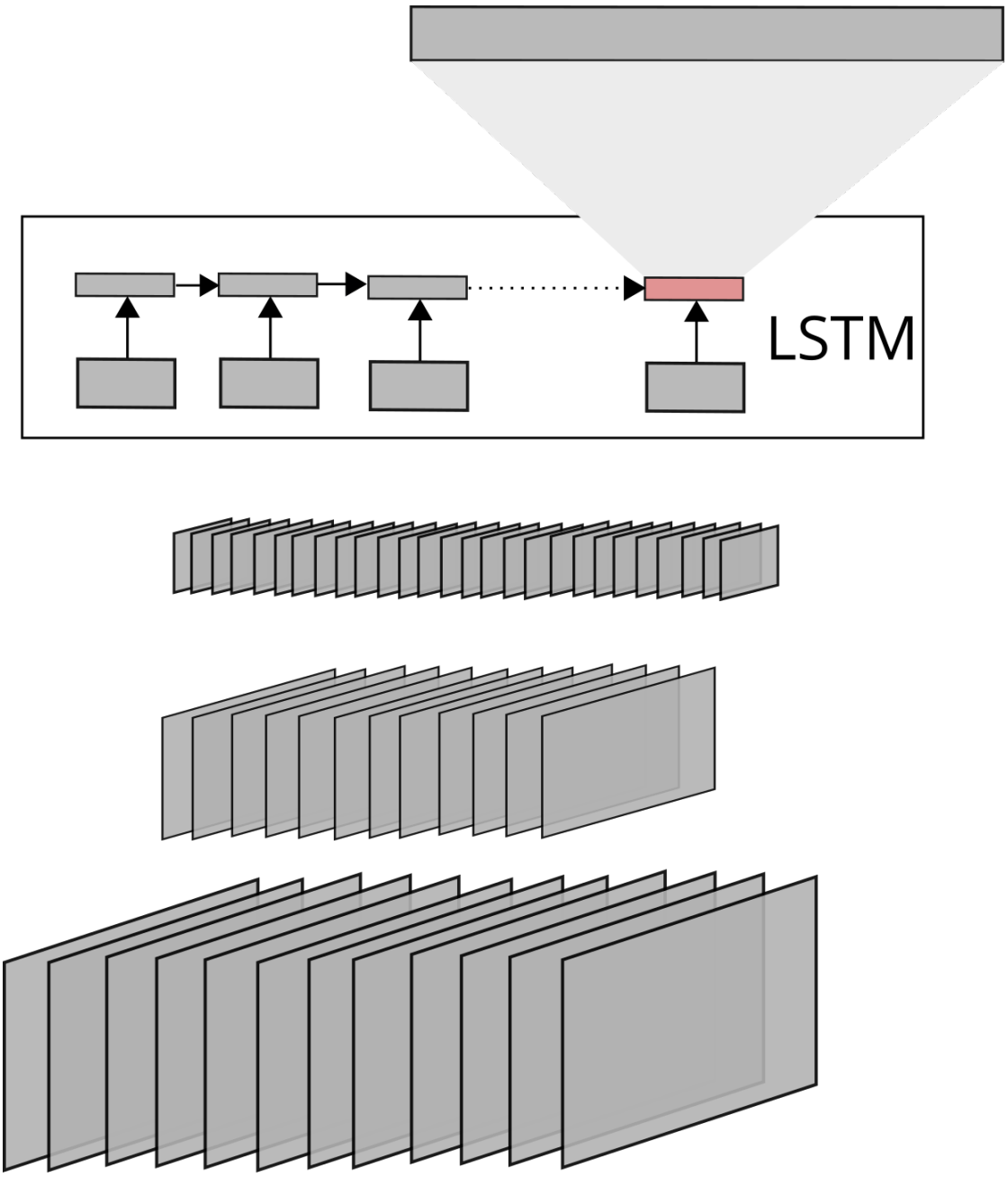

Figure 1:

- Mel-scale

4. Related Work

4.0.1. A UNIFIED DEEP NEURAL NETWORK FOR SPEAKER AND LANGUAGE RECOGNITION

- [17]
- Almost identical: Deep Neural Network Approaches to Speaker and Language Recognition [16]
- high level overview of i-vector system
- Task: Language Recogniton + Speaker Recognition
- use bottleneck feature in the second to last layer
- input 7 static cepstra appended with 49 SDC
- DNN has 7 hidden layers of 1024 nodes each with the exception of the 6th bottleneck layer which has 64 nodes
- LRE11

4.0.2. EXTRACTING DEEP NEURAL NETWORK BOTTLENECK FEATURES USING LOW-RANK MATRIX FACTORIZATION

- [25]
- bottle neck feature improve classification results
- Task: Automatc Speech Recogniton (ASR)
- get bottleneck feature by low rank matrix factorization
- this is done by replacing the usual softmax layer weights by a linear layer with a small number of hidden units followed by a softmax layer
- BN laier is always last layer
- linear layer = FC without activation func

- uses DNN with 5 FCs with 1024 hidden units each + sigmoid activations + 1 BN layer
- softmax cross entropy loss
- 23 critical-band energies are obtained from a Mel filter-bank, with conversation-side-based mean subtraction = 150 dimensions
- further reduciton of output by PCA
- hybrid system of DNN + BN feeding into DNN + BN

4.1. LRE 2015

4.1.1. BAT System Description for NIST LRE 2015

- [15]
- participate in the "Fixed" and "Open" LRE Challenge
- segment data using automated Voice Activity Detection = previously trained NN
- 3042 segments (248 hours of speech) in train set and 42295 segments (146 hours of speech) in dev set.
- inputs 24 log Mel-scale filter bank outputs augmented with fundamental frequency features from 4 different f0 esti- mators
- used i-vector system

4.1.2. Discriminating Languages in a Probabilistic Latent Subspace

- [21]
- Probabilistic Linear Discriminant Analysis (PLDA) model
- In this paper, we review state-of-the-art generative methods, based on the Total Variability (TV) model [10], with the aim to improve their performance with discriminative fine-tuning of each language cluster at a time.

- TV maps audio into single low-dimensional vector, i-vector, that contains speaker, channel, and phonetic variability

4.1.3. Evaluation of an LSTM-RNN System in Different NIST Language Recognition Frameworks

- [24]
- used a one directional LSTM
- perform significantly better than i-vectors systems in LRE
- nice high level description of how LSTMs work
- inputs: random chunks of 2 seconds from which MFCC-SDC (Shifted Delta Coefficients)
- softmax cross entropy loss
- use last frame for scoring
- comparison if i-vector baseline to LSTM
- only used training data for 8 languages with more than 200hours of data
- US English (eng), Span- ish (spa), Dari (dar), French (fre), Pashto (pas), Russian (rus), Urdu (urd), Chinese Mandarin (chi)
- data split into 3, 10 and 30 seconds
- model: two hidden layers of 512 units followed by an output layer. The hidden layers are uni-directional LSTM layers while the output layer is a softmax with as many units as languages in the cluster
- LSTM os only better for short utterance (<= 10s)
- LSTM uses less parameters than i-vector

4.1.4. Frame-by-frame language identification in short utterances using deep neural networks

- [8]
- highlights the downsides/disadvantages of i-vector systems

5. Datasets

In this section we will explain the structure of our datasets and how we obtained them.

Recent breakthroughs in deep learning were fueled by the availability of large-scale, well-annotated, public datasets, for example ImageNet [18] for the computer vision domain. Within the language identification community the TIMIT corpus of read speech [5] has long been the default test set. TIMIT contains a total of 5.4 hours, consisting of 10 sentences spoken by each of 630 speakers from 8 major dialect regions of the United States recorded at 16kHz. Given the short span of each individual sound clip, the overall corpus duration and restriction of only one language it was necessary to obtain our data elsewhere.

This thesis uses two primary datasets collected and processed by us. On the one hand we used speeches, press conferences, and statements from the European Parliament and on the other hand we relied on news broadcasts sourced from YouTube.

5.1. Language Selection

For the scope of this thesis we decide to limit ourselves to a number of high profile languages spoken by many millions around the world. We focused our efforts on languages with a high availability of public speech content present in the various data sources explained below. From a linguistic standpoint we also made sure to include language from within the same language family with similar phonetics for comparison reason. More on that in section 7.2.5. Following these guidelines we decided on two Germanic languages, English and German, and two Romance languages, French and Spanish. We later extended our selection with Russian and Mandarin Chinese.

5.2. EU Speech Repository

The EU Speech Repository[1] is a collection of video resources for interpretation students provided for free by the European Commission. The dataset consists of debates of the the European Parliament, committee press conferences, interviews and tailor-made training material from EU interpreters. All audio clips are recorded in the speaker's native language and feature only one speaker.

With 131 hours of speech data it is the smaller of the two datasets. We obtained material in four languages: English, German, French, and Spanish

[1] `https://webgate.ec.europa.eu/sr/`, accessed 10.03.2017

YouTube Channel Name	Language
CNN	English
BBCNews	English
VOAvideo	English
DeutscheWelle	German
Euronewsde	German
N24de	German
France24	French
Antena3noticias	Spanish
RTVE	Spanish
VOAChina	Mandarin Chinese
Russia24TV	Russian
RTrussian	Russian

Table 1: YouTube channel names used for obtaining the speech data and their corresponding language.

5.3. YouTube News Collection

Following the approach of Montavon [13] we looked for large, public sources of speech audio. We first experimented with podcasts and radio stations both of which are unsuited for the job. Podcasts usually feature only one speaker and radio contains a lot of noise in the form of music. From these initial insights we noticed that news broadcasts provided high quality speech audio data fitting our needs perfectly. To source a large variety of languages and gather enough hours of speech audio we sourced the majority of our data from YouTube.

For each target language we manually selected one or more YouTube channels of respected news outlets. For example for English we used the BBC and CNN to gather a variety of different accents. For a full list of channels refer to table 1. All channels were chosen regardless of their content, their political views or journalistic agenda.

Audio obtained from news coverage has many desired properties. The data is of high recording quality and hundreds of hours recording is available online. News anchors are trained to speak loud and clear, while still talking at a normal conversational speed. News

Feature	EU Speech Repository	YouTube News	YouTube News Extended
Languages	English, German, French, Spanish	English, German, French, Spanish	English, German, French, Spanish, Russian, Chinese
Total audio duration	131h	942h	1508h
Average clip duration	7m 54s	3m 18s	4m 22s
Audio Sampling Rate	48kHz	48kHz	48kHz

Table 2: Comparison of the EU Speech Repository and YouTube News dataset. With ca. 1000 hours of audio recordings the YouTube dataset is ten times large than the EU Speech Repository.

programs often feature guests or remote correspondents resulting in a good mix of different speakers. Unlike speech audio obtained from reading texts aloud, news anchors converse in regular, human, conversational tone with each other. Lastly, news programs feature all the noise one would expect from a real world situation: music jingles, non-speech audio from video clips and transitions between reports. Additionally, the difficulty of our language identification task is increased by mixed language reports. Many city, company, and personal names (e.g., New York City or Google for English) are pronounced in their native language and are embedded within the broadcast's host language. In essence we believe that speech data sourced from news broadcast represent an accurate, real-world sample for speech audio.

In contrast to the EU Speech Repository this dataset consists of ca. 1000 hours of audio data for the same four languages: English, German, French and Spanish. We also gathered an extended language set adding Mandarin Chinese and Russian to the dataset. The extended set is only used for the evaluation of the model extensibility as outlined in section 7.2.8. Table 2 provides a complete comparison between the two datasets.

6. Implementation

This section will outline the software resources of the system, explain the data preprocessing, and describe the model architecture of our neural networks in detail.

6.1. Software

Our language identification system is implemented in Python and uses the open source deep learning framework Keras[3] with the TensorFlow[1] backend for training our neural networks. Keras provides us with a set of higher level machine learning primitives such as convolutional layers and optimizing algorithms without sacrificing any fine grained control over the training parameters. Internally it builds on Google's open source numerical operation library TensorFlow which is optimized to quickly compute multidimensional data on GPUs. We make heavy use Keras' neural network building blocks, e.g. convolutional layers and the efficient LSTM implementation. With the recent announcement of Tensorflow version 1.0[2] it should even be possible to generate a small and efficient binary version of our models ready to be used on mobile phones.

All models are persisted to disk during training, including a summary of the layer architecture as well as the model weights. This makes it easy to load and evaluate model later. During evaluation, the performance metrics are calculated using the Scikit Learn[14] framework. All measurement are logged and visualized using TensorBoard[3], both for the training and validation set. Having the ability to easily study and compare different metrics like accuracy and loss across several training runs made it very comfortable to judge the progress of our research. Figure 2 shows a TensorBoard instance with several models open for discussion.

6.2. Data Preprocessing

All audio files undergo preprocessing before being feed to the neural network. As a first step all files are encoded as uncompressed, lossless Waveform Audio File Format[4], WAVE, commonly know by its file extension *.wav. This conversion has two advantages: A lossless data codec allows for future audio manipulations without any quality loss and makes the data easily readable by third party programs and library such as SciPy[5].

[2] `https://research.googleblog.com/2017/02/announcing-tensorflow-10.html?m=1`, accessed 03.03.2017
[3] `https://www.tensorflow.org/how_tos/summaries_and_tensorboard/`, accessed 30.01.2017
[4] `http://www.microsoft.com/whdc/device/audio/multichaud.mspx`, accessed 23.02.2017
[5] `https://www.scipy.org/`, accessed 23.02.2017

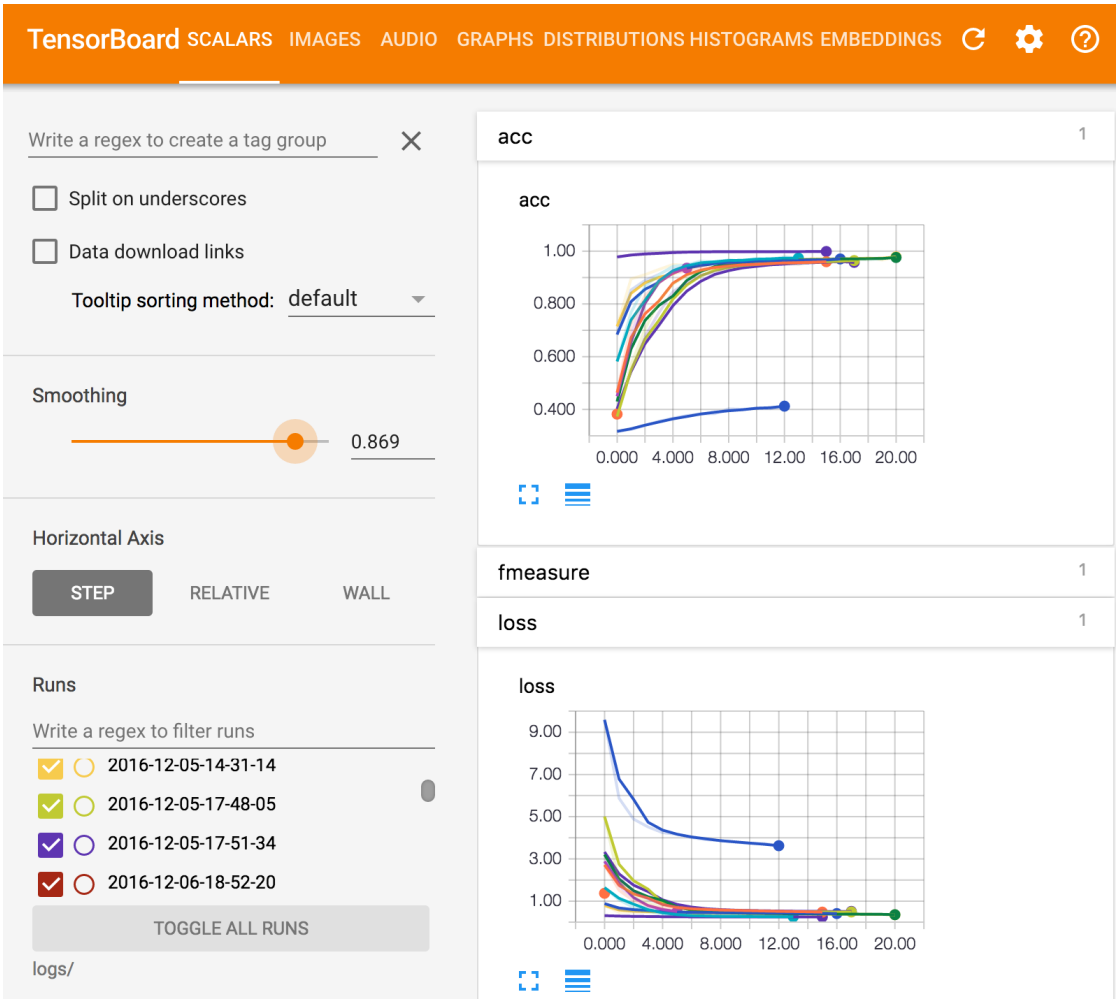

Figure 2: Training accuracy and loss of different models visualized in TensorBoard. Plotting multiple evaluation measures over many training runs helped judge the overall performance progress of the system.

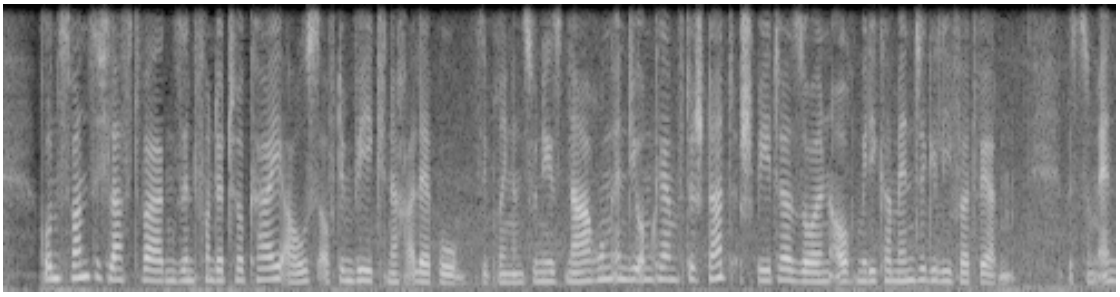

Figure 3: A spectrogram generated from a ten second German audio clip using SoX. Notice the bright ripple-like patterns representing intensive frequency activations. We hypothesize that these frequency activations will serve as the main features for the classifier.

Since our CNN does not operate on raw waveform audio signals directly we transfer our features into the image domain. As introduced in section 3.5 we used a spectrogram representation of the audio file for training our models. The spectrograms were generated using the open source command line tool SoX[6]. Spectrograms are calculated using a Hann window and 129 frequency bins along the frequency axis (y-axis). The time axis (x-axis) is rendered at 25 pixel per second. Each audio sequence is clipped into non-overlapping ten second segments. The final segment is discarded to avoid segments shorter than the required ten seconds. We decided against filtering silent sections within the audio segment to keep the natural pauses between words and not disturb the regular speech rhythm. Frequency intensities are mapped to a gray scale. The resulting greyscale images are saved as lossless PNG files with 500 pixel in width and 129 pixel in height. Appendix A includes the complete listing 1 for generating a spectrogram image with SoX.

As can be seen in figure 3 the spectrograms feature very apparent bright ripple-like pattern. Each of these represents a strong activation of a certain frequency over time. Several frequency activations can be active simultaneously constituting a particular phoneme or sound. A sequence of these phonemes forms words and is only interrupted by short speech pauses. We hypothesize that our LID system will learn the characteristic and unique composition of these frequency activation for every language in our classifier.

6.3. CNN Architecture

To successfully solve our research task with deep neural networks we had to design a fitting model architecture. Combing the right layers and adjusting the correct parameters is not an easy task. Critics argue that deep learning systems are a black box and that the impact of individual settings is hard to judge. Therefore, we based our model architectures on

[6] `http://sox.sourceforge.net/`, accessed 23.02.2017

proven existing designs from related work and adapted them to our needs, while heeding as best practices[22][12].

The size of the deep neural network is determined by the bias-variance tradeoff[6], imposing underfitting or overfitting on the resulting model. The bias characterizes the prediction error that results from the model's limited ability to learn relevant relations of features in the training data. Underfitting can be caused by designing a too small network layer causing a high bias. Variance in contrast refers to the error that results from the model responding overly sensitive to variation in the training data. By designing a too large network and introducing too many parameters the model results in low variance. Hence, it learns an exact representation of the training data without abstracting for more general applications. This is known as overfitting. Designing and adjusting the network layout is an empirical process. We tried many variations and parameters of our proposed model layouts to find the most suitable design for our LID system. The layout of the network interdepends and interacts with other design decisions of the system, especially the loss calculation. It hence needs to be tested in a complete process setup.

For this thesis we tried two different convolutional neural networks. The first one is based on the early VGG-style model architectures of Simonyan et al. [2]. With only five convolutional layers and modest feature map output sizes this is the smaller of two designs. Note the large kernel size of 7x7 pixels and 5x5 pixels for the first two layers yielding a large receptive field. We complemented each convolutional layer with batch normalization[9], a technique that both helps with training speed and model regularization. The full network layout is available in table 3.

The second CNN architecture is based on the VGG-16 network[20] and has considerably more parameters. The CNN is deeper with seven convolutional layers and has increased feature map outputs in an effort to capture more information. In contrast to the first design, all convolutional layers feature equally sized kernels of 3x3 pixels. The complete model architecture can be found in table 4.

6.4. CRNN Architecture

Write something here.

Layer Type	output size	kernel	stride
Convolution with Batch Normalization	123 x 494 x 16	7x7	1
Max Pooling	61 x 247 x 16	2x2	2
Convolution with Batch Normalization	57 x 243 x 32	5x5	1
Max Pooling	28 x 121 x 32	2x2	2
Convolution with Batch Normalization	26 x 119 x 64	3x3	1
Max Pooling	13 x 59 x 64	2x2	2
Convolution with Batch Normalization	11 x 57 x 128	3x3	1
Max Pooling	5 x 28 x 128	2x2	2
Convolution with Batch Normalization	3 x 26 x 256	3x3	1
Max Pooling	1 x 13 x 256	2x2	2
Dropout & Flatten	3328		
Fully Connected	1024		
Fully Connected	4		

Table 3: The layerwise architecture for the convolutional neural network CNN_A.

Layer Type	output size	kernel	stride
Convolution with Batch Normalization	127 x 498 x 64	3x3	1x1
Max Pooling	63 x 249 x 64	2x2	2x2
Convolution with Batch Normalization	61 x 247 x 128	3x3	1x1
Max Pooling	30 x 123 x 128	2x2	2x2
Convolution with Batch Normalization	28 x 121 x 256	3x3	1x1
Convolution with Batch Normalization	26 x 119 x 256	3x3	1x1
Max Pooling	13 x 59 x 256	2x2	2x2
Convolution with Batch Normalization	11 x 57 x 512	3x3	1x1
Convolution with Batch Normalization	9 x 55 x 512	3x3	1x1
Max Pooling	4 x 27 x 512	2x2	2x2
Convolution with Batch Normalization	2 x 25 x 512	3x3	1x1
Max Pooling	1 x 12 x 512	2x2	2x2
Flatten	6144		
Fully Connected	1024		
Fully Connected	4		

Table 4: The layerwise architecture for the convolutional neural network CNN_C.

Layer Type	output size	kernel	stride
Convolution with Batch Normalization	123 x 494 x 16	7x7	1x1
Max Pooling	61 x 247 x 16	2x2	2x2
Convolution with Batch Normalization	57 x 243 x 32	5x5	1x1
Max Pooling	28 x 121 x 32	2x2	2x2
Convolution with Batch Normalization	26 x 119 x 64	3x3	1x1
Max Pooling	13 x 59 x 64	2x2	2x2
Convolution with Batch Normalization	11 x 57 x 128	3x3	1x1
Max Pooling	5 x 56 x 128	2x2	2x1
Convolution with Batch Normalization	3 x 54 x 256	3x3	1x1
Max Pooling	1 x 53 x 256	2x2	2x1
Permute	53 x 1 x 256		
Reshape	53 x 256		
Bidirectional LSTM	1024		
Fully Connected	4		

Table 5: The layerwise architecture for the convolutional recurrent neural network.

7. Experiments and Evaluation

In this chapter we show and discuss the results of training the outlined neural network architecture for spoken language identification. We introduce several performance metrics and present the results evaluated on our system. Further, we experiment with modified model architectures to optimize our model for noise robustness. To assess the real world performance of the the LID system we augment our data to simulated various noisy environments. We show the classification performance of our approach and discuss the system's inter language discrimination and extensibility to other languages.

7.1. Setup

7.1.1. Hardware Resources

In order to facilitate Keras' and TensorFlow's hardware-accelerated computation we executed all trainings on CUDA[7] compatible GPU machines. All experiments were executed on two CUDA enabled machines belonging to the Internet Technologies and Systems chair. Details can be found in table 6.

	Machine A	Machine B
OS	Ubuntu Linux 14.04	Ubuntu Linux 16.04
CPU	Intel® Core™ i7-4790K @ 4GHz	AMD FX™-8370 @ 4GHz
RAM	16GB	32GB
GPU	Nvidia GeForce® GTX 980	Nvidia Titan X
VRAM	4GB	12GB

Table 6: Hardware resources used in training the neural networks.

7.1.2. Data

For our performance evaluation we used the European Speech Repository and YouTube News dataset as describe in section 5. Both datasets were preprocessed and converted to spectrogram images as described in section 6.2. Each spectrogram image represents a non-overlapping ten second duration of source audio file. We split both dataset into

[7] `https://developer.nvidia.com/cuda-zone`, accessed 30.01.2017

	European Speech Repository	YouTube News
Training Set	18.788	193.432
Validation Set	5.372	55.272
Test Set	2.684	27.632
Total	26.844	276.336

Table 7: The amount of samples for our training (70%), validation (20%) and testing (10%) set for the respective datasets.

a training (70%), validation (20%) and testing set (10%) and all files were distributed equally between all four language classes. The number of samples per class were limited by the language with least files to ensure an equal class distribution. The European Speech repository yields a total of ca. 19.000 training images files which amounts to roughly 53 hours of speech audio. The YouTube News dataset is considerable larger and yields a total of ca .194.000 training files, or 540 hours. Table 7 contains the detailed dataset splits.

Given the European Speech Repository's smaller size we only used it initially confirm the validity of our approach. Since we were satisfied with the results we did not do include it in the extensive robustness tests that we used for the evaluation on the YouTube News dataset. In addition to the original audio we augmented the news dataset with three different background noises to evaluate how well our model would hold out in non ideal, real world situations outside of a new broadcasting studio. For the first experiment we added generic white noise to data. For the second experiment we added noise to simulate an old phone line or bad internet connection during voice chat. Lastly, we added background music to the data. All experiments are described in detail below.

7.1.3. Training of the Neural Network Model

Neural networks have a multitude of hyperparameters that influence the training results drastically. In this section we will briefly explain our choice of hyperparameters and other important training settings.

Optimizer We employed an Adam[10] optimizer to quickly and efficiently convergence our model. The Adam solver utilizes momentum during gradient updates to support a quicker convergence. We set the optimizer's parameters β_1 to 0.9, β_2 to 0.999, and ϵ to 1e-08. Overall we found it to be an order of magnitude quicker than using standard stochastic gradient descent (SGD). We resorted to SGD during finetuning

when we needed more control over the learning rate schedule and wanted smaller weight updates.

Weights Initializer All layer weights are initialized within the range [0, 1) using Keras' default random Glorot uniform initializer[7].

Data Normalization The greyscale images are loaded using SciPy and normalized to the [0, 1] range. The shape for all inputs needs to be uniform across all samples and is set to [500, 129, 1], unless otherwise noted. The data loader uses Python generators[8] to keep the system's memory footprint low.

Learning Rate We set the initial learning rate to 0.001. Given the Adam optimizer's dynamic learning rate adaption we expect the learning rate to be increase or decreased after every epoch.

Batch Size We specified the batch size depending on the available VRAM of the training machine. We used a value of 64 for Machine A and 128 for Machine B. See section 7.1.1 for the hardware specifications.

Weight Regularization We employed the L2 norm as a weight regularizer for all convolutional and fully connected layers to improve the models generality. We penalize our loss with a weight decay value of 0.001. Additional regularization happens through the use of Batch Normalization layers.

Epochs We limited the model training to a maximum of 50 epochs when using the Adam solver. We usually reach convergence well below this threshold. For training sessions with SGD we increased this parameter considerably. To speed up our workflow we employed an early stopping policy and stopped a training if the validation accuracy and loss did not increase within a ten epoch window.

Metrics We observed the loss, accuracy, recall, precision, f1 measure, and equal error rate for both the training and validation set during model training. All values were saved to log files and visualized as graphs in TensorBoard.

Loss As is common for multivariate classification all models were trained with a softmax cross-entropy loss function.

[8] `https://docs.python.org/3/glossary.html#term-generator`, accessed 30.01.2017

7.2. Evaluation

7.2.1. Evaluation Metrics

In this section we discuss the evaluation metrics used throughout our experiments. All metrics are generally only defined for binary classification results. Given our multi-class classification problem we will report the average of the individual class performance measures in the following sections.

Accuracy is a common measure in machine learning and is defined as the ratio of correctly classified samples to all samples in the dataset. In the context of language identification this translates as:

$$accuracy = \frac{\left|\{\text{correctly identified language samples}\}\right|}{\left|\{\text{all language samples}\}\right|}$$

Precision and Recall Precision defines the ratio of retrieved language samples that are correctly identified as belonging to said language. Recall is the fraction of correctly identified language samples to all samples belonging to this language.

$$truePositives = \left|\left\{\begin{matrix}\text{samples belonging to a language which were correctly identified} \\ \text{as belonging to said language}\end{matrix}\right\}\right|$$

$$falsePositives = \left|\left\{\begin{matrix}\text{samples belonging to a language which were identified as be-} \\ \text{longing to another language}\end{matrix}\right\}\right|$$

$$falseNegatives = \left|\left\{\begin{matrix}\text{samples belonging to a language which were incorrectly identi-} \\ \text{fied as not belonging to said language}\end{matrix}\right\}\right|$$

$$precision = \frac{truePositives}{truePositives + falsePositives}$$

$$recall = \frac{truePositives}{truePositives + falseNegatives}$$

The F1 Score is the scaled harmonic mean of precision and recall. It is used to a have combined judgement of recall and precision, since one is generally not interested in assessing one without the other.

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

7.2.2. Results for EU Speech Repository Dataset

In order to verify our theoretic model of using convolutional neural networks for classifying audio data in the image domain we established a baseline with the smaller EU Speech Repository dataset. Previous work with CNNs showed that the careful arrangement of the neural network layers has a great effect on the final classification performance. If one does not use enough or sufficiently large layers the model is not able to properly distinguish between classes. Going too deep or using too large layer outputs increases the overall number of model parameters to a degree where both training time and classification performance suffers again. The goal of this experiment is to find favorable settings for the number of convolutional layers needed, the kernel size of the convolutions, the number of output maps of the convolutional layers and finally the number of features of the fully connected layer.

For this particular dataset we tested three slightly different model architectures. Following the VGG-style model architecture of Simonyan et al. [2], we setup a first CNN with five convolutional layers as outlined in section 6.3. The first two convolutional layers use larger kernel sizes of 7x7 and 5x5, respectively. All following kernels were set at 3x3. Each convolutional layer is followed by batch normalization and a 2x2 pooling with a stride of two. After the five convolutional blocks we add regularization through a fifty percent dropout before flattening all parameters to a fully connected layer with 1024 outputs. The final fully connected layer serves as a classifier outputting the prediction for our language identification. Henceforth, we will refer to this model as CNN_A.

A slightly adapted version called CNN_B has the same amount of convolutional layers but with a reduced number of feature maps. Instead of doubling the initial value of sixteen feature maps for every convolutional layer we stick to a schedule of 16 - 32 - 32 - 64 - 64 feature maps, respectively. The fully connected layer has also been reduce to only 256 output units. Overall this model has significantly less parameters than the CNN_A. The purpose of this variation is to ensure that the proposed architecture for CNN_A is not unnecessarily complex.

Lastly we evaluated architecture CNN_C which uses constant kernel size of 3x3 for all convolutional layers. At the same time we increased the number of convolutional layers to

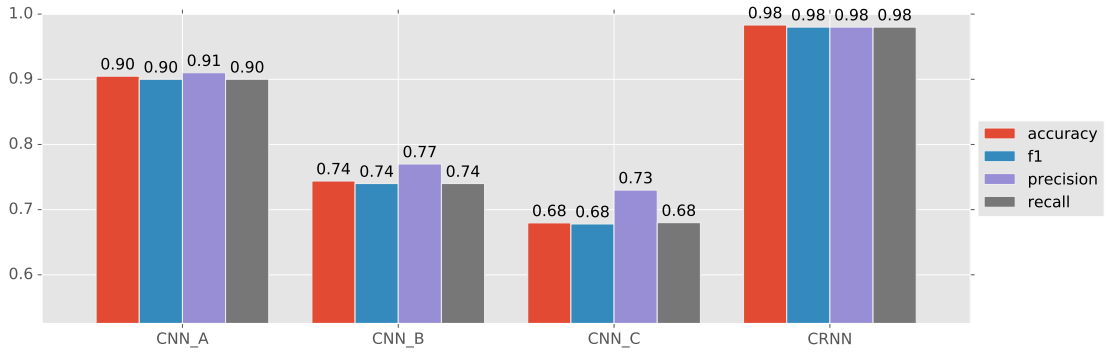

Figure 4: Performance measure comparison of three different CNN architectures evaluated on the EU Speech Repository dataset and our proposal of a CRNN model. CNN_A outperforms all other CNN approaches with a top accuracy of 0.90, but is bested by the CRNN's 0.98 accuracy, proving the potential of this thesis' approach.

seven and extended feature maps for each layer: 64 - 128 - 256 -256 - 512 - 512 - 515. The remaining fully connected layers are identical to the CNN_A. The main difference here is the smaller receptive field of the convolutional layer which could negatively effect the model performance but should speed up the model training time overall. The complete CNN_C architecture is laid out in table 4.

CNN_A outperforms both of the other two network architectures with respect to all the evaluated performance measures, cf. figure 4. With a top-1 accuracy of 0.904 it trumps CNN_B and CNN_C with 0.743 and 0.68, respectively. Comparing the F1 score we get a similar result: 0.90 versus 0.74 and 0.68. This experiment confirmed a few of our initial hypotheses. Firstly, it proves that convolutional neural networks can be successfully used to classify audio data. Secondly, it demonstrates that spectrogram images are meaningful representation for audio that retains enough information for language identification. Thirdly, it shows that large kernels for the initial convolutional layers are indeed favorable. The increased receptive field captures both the time domain and frequencies better. Based on these findings we did some further testing with CNN_A. Based on Mishkin et al.[12] we switched the convolutional layers' ReLU activations to Exponential Linear Units[4] (ELU) but without any improvement. Baoguan et al. [19] proposed to use 1x2 rectangular pooling windows instead of the conventional square ones. This tweak yields feature maps with larger width, hence longer features along the time domain. In theory this should increase the CNN's sensitivity at judging the occurrence of frequencies at certain time intervals. For this experiment we were unable to gain any improvement, but we will discuss this approach some more for our CRNN approach later.

The goal of this thesis is to evaluate the use of Deep Convolutional Recurrent Networks. Therefore we extended our previously best performing CNN_A with a bidirectional

LSTM layer. We interpreted the CNN output as intermediate representation of the audio frequencies and used every vector entry along the x-axis as a single step / input for the LSTMs as explained in section 3.4. During the training we froze the convolutional layers to only train the LSTMs weights and learn the frequency sequence of the audio sample. Our bidirectional LSTM layer trained two individual LSTMs with 512 outputs each, one training the input sequence from the start and one from the end. Both outputs were concatenated to form a single output vector with 1024 dimensions which is followed by single fully connected layer for classification. Our CRNN architecture outperformed all CNN approaches significantly. With a top-1 accuracy of 0.98 and a F1 score of 0.98 it proves the viability of the CRNN approach and reaffirms the central hypothesis of this thesis.

7.2.3. Effect of Audio Duration

For all previous experiments we split all audio recording into ten second segments, which translated into an image dimension of 500x129 pixels for the spectrogram. We decided on 10 second audio snippets based on the setup of the NIST LRE 2015 [9] challenge. To study the effect of the audio duration on classification performance we set up a version of the EU Speech Repository dataset with non-overlapping five seconds snippets but left the number of frequency bins unchanged. Hence we reduced the input dimensions to 250x129 pixels and could not just use our previously trained models but had to retrain new models.

To set a baseline we used the same CNN_A architecture as explained in the previous section. When completely training it from scratch we achieved an accuracy of 0.81 falling short of the results achieved with ten second snippets. Next, we applied some transfer learning and finetuned CNN_A on the new five second dataset. Since the convolutional layers are not bound to a specific input size and given that the frequency features did not change in dimension we were able to reuse the complete convolutional weights. For the finetuning we froze the convolutional layers, confident in their ability to detect frequencies, and only retrained the final two fully connected layers. With the bisection of the input data the amount of model parameters were greatly reduced, especially the fully connected layer weights. To account for this we finetuned one model with a fully connected layer of 512 outputs and a second one with the default 1024 outputs. Overall this yielded an accuracy of 0.88 and 0.89, respectively. We concluded that the effect of the smaller fully connected layer is only marginal.

After establishing a solid CNN foundation we applied the same CRNN approach as previously highlighted. Due to the shorter audio duration the final pooling layer only features 5 output units along the x-axis compared to the 13 output units of the ten second

[9] `https://www.nist.gov/itl/iad/mig/2015-language-recognition-evaluation`, accessed 15.02.2017

Model Architecture	Accuracy	F1
CNN from scratch	0.81	0.81
CNN finetuned (FC 1024)	0.88	0.89
CNN finetuned (FC 512)	0.89	0.89
CRNN with 5 time steps	0.90	0.91
CRNN with 22 time steps	0.90	0.91
CRNN (10s) for reference	0.98	0.98

Table 8: Various CNN and CRNN model configurations trained on five second audio samples. The best performing five second CRNN still falls short of its ten second counterpart with an accuracy of 0.90 and 0.98, respectively.

CRNN. When interpreted as a sequence of time steps and fed into the bidirectional LSTM the accuracy improved only marginally to 0.90. We suspect that the number of time steps was too little to take full advantage of the recurrent network layer. In an effort to increase the number of output units of the final pooling layer and hence increase the sequence length we applied 1x2 rectangular pooling tweak again. We changed the final two pooling layers and bumped up the number of output units along the x-axis to 22. The y-axis remained unaffected. The resulting accuracy of 0.90 and the F1 score of 0.91 remained comparable to the previous model and did not bring the desired effect.

In summary we believe that decreasing the duration of the audio snippets used for training has a negative effect on the classification performance. While it is possible to train and finetune CNNs that match their ten second counterparts we found that the CRNN approach does not boost the model effectiveness in a similar manner.

7.2.4. Results for YouTube News Dataset

Following the promising results from the experiments with the EU Speech Repository we switched to the ten time larger YouTube News dataset for further training and evaluation. For our first experiment we used the same CNN_A architecture as before but initialized the model weights with the weights of best performing model of the EU Speech Repository evaluation. Our reasoning here was to reuse the convolutional layers that were already trained to identify frequency ranges. However, with an accuracy of only 0.79 it did not perform as strongly as anticipated. One reason for this could be that the EU dataset is a lot smaller and does not feature as many diverse situation as present in news broadcasts since all audio is recorded in a similar environment without much background noise and

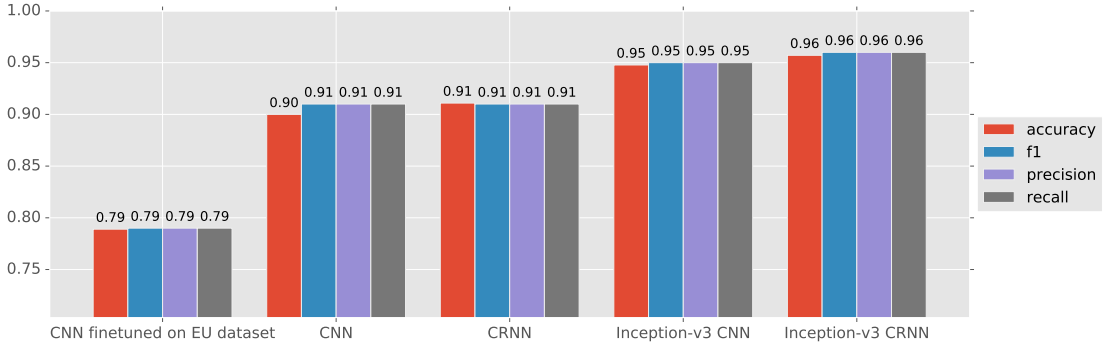

Figure 5: Performance measurement comparison between our CNN and CRNN models and Inception-v3 based models. With a top accuracy of 0.96 the Inception style CRNN performs best but needs more than five times the amount of parameters compared to our proposed model architecture.

exhibits high signal quality.

Next, we trained the same CNN completely from scratch with randomly initialized weights. We had several setbacks when using an Adam optimizer and reverted to using standard SGD to keep our loss in check. Additionally we employed gradient clipping to avoid the exploding gradient problem. With these measure we were able to get the model to converge and gained an accuracy of 0.9090 besting our previous attempt. Given the new dataset larger size we also tried to increase the number of parameters by doubling the feature maps of the convolutional layers. This, however, did not help. As a next step we applied the CRNN approach again. Based on this CNN we added our bidirectional LSTM layers in the same manner as for the previous CRNNs and were able to slightly improve our accuracy and F1 score to 0.9124, respectively.

To evaluate how our model architecture fares against established deep learning models we trained a model using Google's Inception-v3[22] layout. This resulted in a top accuracy of 0.9488 improving our results significantly. Applying the established CRNN treatment to this model increased the performance by roughly 1% to an accuracy of 0.9579. In order to boost the performance even further we also tried a CRNN variation where both the convolutional layer weights and the LSTM weights were trained. In contrast to our initial CRNN approach this method also updates the existing convolutional filter. We found, however, that this variation did not improve performance. Figure 5 shows an overview of the performance metrics of the mentioned experiments. The increased performance, however, does not come without a cost. With a total of 3.153.924 parameters our CRNN uses roughly six times less parameters then the Inception-v3 CRNN with its 19 million. This increases training time, requires more training data and consumes more GPU memory. On disk the serialized model weights come in at 30MB versus 260MB, which could be a potential disadvantage for deployment on mobile phones.

	EN	DE	FR	ES
EN	6153	339	181	225
DE	426	6128	173	162
FR	200	145	6447	107
ES	214	170	115	6399

Table 9: Confusion matrix of the best performing CRNN. English audio files are likely to be misclassified as German and vice versa. Both languages belong the family of Germanic languages.

Plot trainings loss as a space filler?

7.2.5. Inter Language Discrimination

Previous work[13] raised concerns about the similarity of our four feature languages – English, German, French and Spanish – and the model's inability to discriminate between them properly. Both English and German belong to the West Germanic language family, while French and Spanish are part of the Romance languages. We hypothesized that our deep learning approach to language identification is able to differentiate between them reliably. Table 9 shows the confusion matrix when evaluating language family pairs on the best performing CRNN. Spanish and French audio files separate very well with hardly any wrong classifications. Both languages are more likely to be classified as German and English rather then as the respective other, demonstrating that their learned representations are quite distinctive within their language family. German and English language samples have a tendency to be misclassified as the respective other. Furthermore English also has a slight bias towards French, an observation in line with related work[23]. German, however, distributes its classification error evenly between French and Spanish. Overall, German samples are misclassified the most across all languages.

The confusion matrix for the Inception-v3 CRNN, table 10, match our observations for the other model. Again, German exhibits the most classification errors.

7.2.6. Noise Robustness

Given that we left our data unadulterated we expect a certain degree of noise within the dataset. Therefore we hypothesize that the neural network develops some noise robustness by itself. For instance, the convolution operations of the earlier layers summarize pixel

	EN	DE	FR	ES
EN	6648	140	45	61
DE	152	6639	62	44
FR	68	59	6742	27
ES	75	83	42	6697

Table 10: Confusion matrix of the Inception-v3 CRNN. Despite its deeper architecture it makes the same mistakes as our proposed CRNN.

values over an image patch and help with masking noise. To prove our theory we generated two augmented datasets based on the YouTube news dataset. For the first one we mixed the audio signal with randomly generated white noise sound. The resulting audio samples are still easily identifiable for human listeners. The noise has a very strong audible presence, so much so that a human tester will easily be annoyed after a few seconds of listening to it. For the second augmentation we added a more periodic cracking noise emulating analog telephony or a bad voice chat connection. We sampled a selection of fitting sound effects and randomly applied these to the source signal using the PyDub[10] library. The resulting noise is not as noticeable and less intrusive as the white noise, but gives the augmented audio file a subdued vintage quality.

The white noise did deteriorate the language identification performance significantly both for our CRNN proposal and the Inception-v3 CRNN as can be seen in table 11. The noise spectrogram in figure 6 show that the white noise effect has a very strong influence on the resulting image. Most parts of the image end up being covered by the distinct noise texture and only the lower frequency features remain intact. All pauses and fine grained details are lost to the noise sound. The cracking experiment fared did not incur such a dramatic drop in performance. That might be in part due to the consistent recurring sound of the noise in contrast to the randomly generated white noise sound. Perhaps a second factor was the lower, less intrusive volume used for augmenting this dataset.

The deeper, more complex structure of the Inception-v3 CRNN did suffer a significantly smaller performance deterioration than our proposed CRNN model architecture. The more than five times as many parameters seem to capture the frequency features in a more robust manner. In an attempt to remedy the performance loss for our model we trained and finetuned models containing white noise data. We experimented with a 100% noise dataset and the original news dataset extended with 10% white noise for augmentation purposes. While both approaches did recovered some white noise performance they reduced the general language identification and cracking noise statistics as a trade off. Let it be noted that our audio mixing was fully automated and that the resulting samples

[10] `https://github.com/jiaaro/pydub`, accessed 01.03.2017

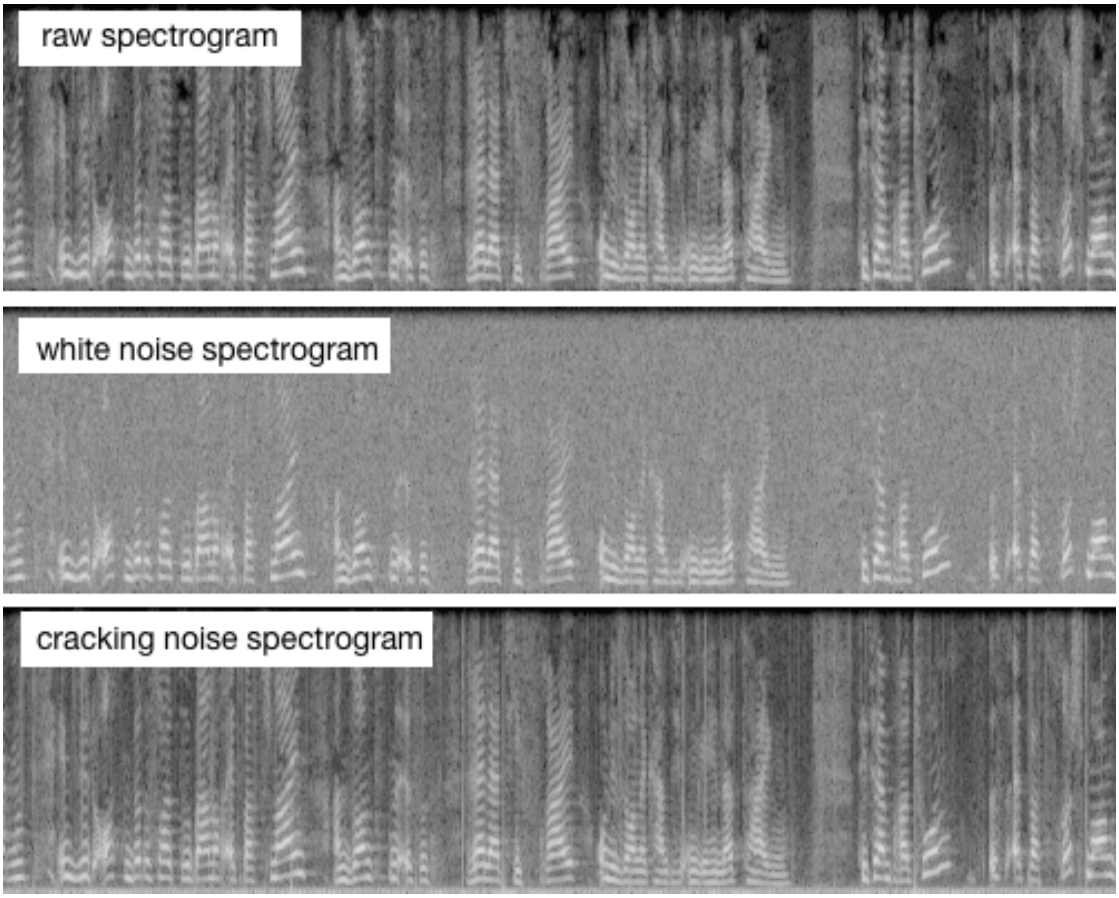

Figure 6: Spectrograms generated from the raw data, augmented with white noise and mixed with a cracking noise emulating analog telephony or a bad voice chat connection. The white noise aggressively subdues most higher frequencies and pauses causing a loss of classification performance. The cracking noise is less intrusive and therefore does not affect accuracy as much.

varied in speech and noise volume. We tried to match volume levels of speech and noise to maintain as much clarity of speech as possible, yet some samples will have an artificial quality to them.

Overall we note that even deep convolutional recurrent networks are still subject to the influence of noise on audio. We learned that our spectrogram preprocessing does not help in these situations and that different network architectures play an instrumental role in dealing with these situations.

Dataset	CRNN		Inception-v3 CRNN	
	Accuracy	F1	Accuracy	F1
white noise	0.63	0.63	0.91	0.91
cracking noise	0.82	0.83	0.93	0.93

Table 11: Accuracy and F1 score for our models evaluated on the speech data augmented with two different types of noise.

7.2.7. Background Music Robustness

Many real world audio applications involve a some form of music. Therefore we evaluated our model to see if language identification was still possible when mixing speech with music. For this experiment we used two different test sets. For the first series we augmented our existing YouTube news dataset with randomly sampled background music in a similar fashion to the background noise augmentation. The background audio was obtained from Soundcloud[11] and features royalty free, instrument-only music from various genres, including Pop, Dubstep, Rock and Electro amongst others. We normalized the volume of these sounds and overlaid them onto our speech data while trying to stay below the speech audio volume. In the best cases the two audio streams blend nicely with the background music producing a soft but noticeable ambient effect, while retaining the clarity of the speech. In other cases the audio levels and volume are over-accentuate for one or the other. The final result, however, does neither resemble a song nor a piece of music. We changed neither the tempo nor the pitch of our speakers and the rhythm of the vocals does not match the rhythm of the background audio. Therefore we gathered a small second test set of XXX songs for each of the following genres: Pop, Rock, Hiphop, and Folk. Given that our training set does not intentionally include samples with music or songs we expected the performance do be drastically lower then for pure speech samples.

insert number of songs

7.2.8. Model Extensibility

So far all our experiments were conducted on datasets consisting of only four languages: English, German, French and Spanish. We complemented these with two new languages spoken by millions around the globe: Mandarin Chinese and Russian. The goal of this experiment was to learn whether we could easily expand our model to other languages. We increased our existing YouTube news dataset with samples taken from Chinese and Russian news channels yielding the extended YouTube news dataset described in section 5.3. In order to maintain the class distribution for six languages we had to decrease the

[11] `https://soundcloud.com/royalty-free-audio-loops`, accessed 01.03.2017

Dataset	CRNN		Inception-v3 CRNN	
	Accuracy	F1	Accuracy	F1
Background Music	0.70	0.70	0.89	0.89
Pop Songs	0.XX	0.XX	0.XX	0.XX
Rock Songs	0.XX	0.XX	0.XX	0.XX
HipHop Songs	0.XX	0.XX	0.XX	0.XX
Folk Songs	0.XX	0.XX	0.XX	0.XX

Table 12: Accuracy and F1 score for our models evaluated on the speech data augmented with background music and songs sampled from four different genres.

number of training samples of the existing samples slightly. Table 2 contains the details for the extended YouTube news dataset.

For this experiment we first finetuned our previous best CNN by replacing the final fully connected layers and adjusted the number of output nodes to accommodate for six classes. The resulting model served as the basis for training the CRNN in a similar manner as in earlier experiments. Applied on the test set we measured an accuracy of 0.92 and F1-score of 0.92. Both measurements match our previous evaluation with four languages on the YouTube news dataset as laid out in section 7.2.4 proving that the proposed CRNN architecture can easily be extended to cover more languages. Figure 7 shows individual performance measure of each language. Mandarin Chinese outperforms all other languages with a top accuracy of 0.96, which could be interpreted as it sounding the most contrasting to western languages and featuring its own unique intonation. We also noted that Russian was most frequently misclassified as Spanish and vice-versa. In contrast to our previous observation German is no longer the worst performing class, but English takes that role now. This is in part due to a significant number of misclassification as Russian samples.

Include the table or is the plot enough?

Given that both new languages are rooted within their own respective language families and feature considerable different intonations we were content to find that the features learned by our model are indeed universal in nature. We are confident in the believe that the approach to language identification proposed in this thesis can be successfully applied to a wide variety of languages.

	accuracy	precision	recall	F1
EN	0.86	0.89	0.88	0.88
DE	0.89	0.91	0.90	0.90
FR	0.93	0.94	0.93	0.94
ES	0.92	0.91	0.93	0.92
CN	0.96	0.97	0.96	0.96
RUS	0.92	0.90	0.92	0.91

Table 13:

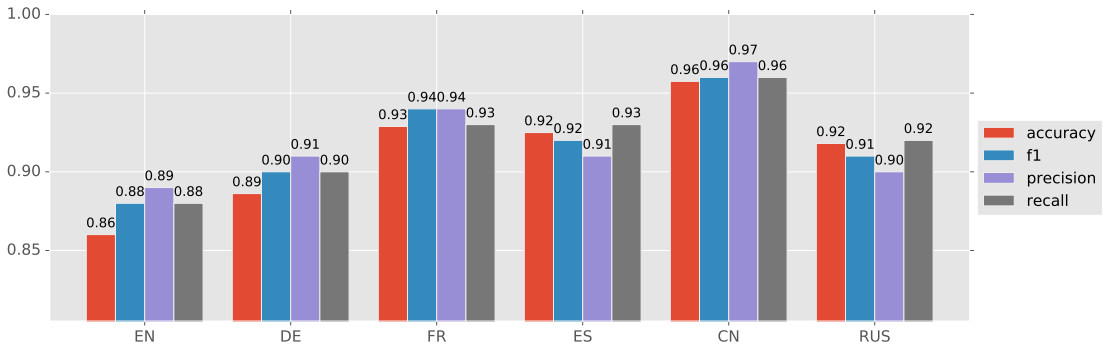

Figure 7: Individual performance measurements for each of our six target languages: English, German, French, Spanish, Mandarin Chinese, and Russian. Chinese exhibits the lowest misclassification rate while English performs the worst. Overall the model performance is consistent with previous evaluations on four languages as highlighted in section 7.2.4.

7.2.9. Visualizations

All previous sections described and evaluated our model by measuring various performance indicators and applying them to different datasets. In this section we will present plots underlining earlier observations from a different perspective.

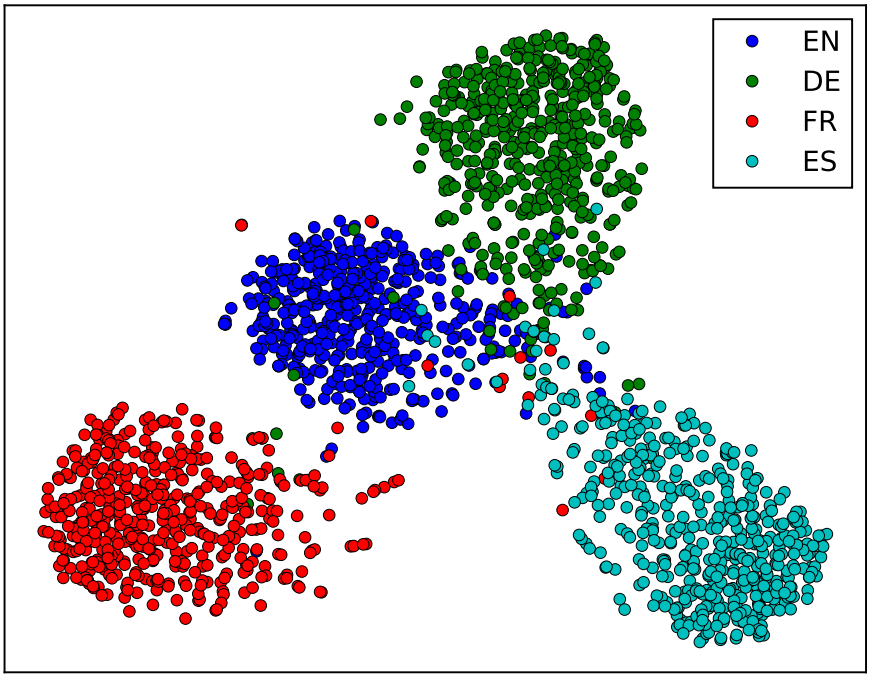

Figure 8: A 2D t-SNE plot of the high dimensional vector representation of our YouTube news samples. All four language classes form distinct clusters confirming the network learned effective representations of the audio features.

First we visualized the high dimensional language vector embedding space using the t-distributed stochastic neighbor embedding (t-SNE) algorithm[11]. t-SNE is a nonlinear dimensionality reduction technique employed to map high dimensional data into 2D or 3D space for plotting. We applied this machine learning algorithm to the first 2000 predictions of our second to last fully connected layer right before the classifier and managed to project our 1024 dimensional YouTube news data into a 2D space. Figure 8 shows the resulting plot highlighting a good separation of our four language classes as independent clusters confirming the network learned effective representations of the audio features. Note that French and Spanish split very nicely while German and English have some

overlap. This is in line with our previous observations of classification errors as described in section 5.3.

A primary advantage of deep neural networks is their ability to identify and learn useful features on their own making them powerful and versatile. From an end user's perspective they can appear as a bit of a black box and it remains unclear which features they ultimately deem relevant. In order to get a better understanding of our model we visualized its convolutional layers. In figure 9 we visualized nine of the highest activating filters of the final convolutional layer. In order to obtain these filters we performed back propagation from the output of the filter back to an input image. This yielded the gradients of the output of the filter with respect to the input image pixels. We used that to perform gradient ascent, searching for the image pixels that maximize the output of the filter.

Visualizations for the lower level convolutional layers resulted in images of very simple geometric shapes like lines and circles which matched our expectations. With increasing network depth each convolutional layer's features combined and evolved into more complex shapes resembling our input domain. In figure 9 we can identify the familiar ripple like patterns that form frequency activations over time. This proves that the network learned these structures as we hypothesized earlier. We can also identify that some filters specialize in high frequency whereas others focus on low frequencies. Furthermore, it can be observed that the filters only react to a short and specific span of time within the spectrogram, all of which are less than one second in duration.

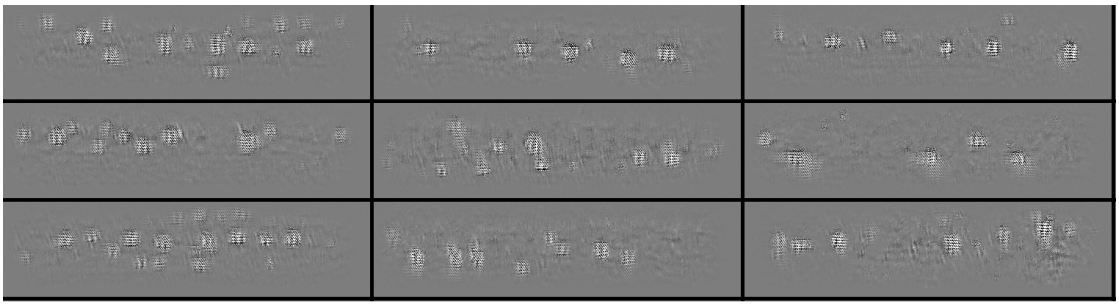

Figure 9: Visualization of nine filters of the final convolutional layer. Note the ripple like patterns responsible for detecting different frequency spans in the input audio.

7.2.10. Discussion

In this section we introduced and compared different CNN and CRNN architectures for language identification and proved that these models can be used to accurately solve our research task. We showed that these results hold true regardless of the input audio source and across various languages. This thesis also confirms that audio tasks can be solved

within the image domain using spectrogram image representations. As hypothesized we were able to prove that our system learned the audio and frequency structures of these spectrogram images. We demonstrated that the learned intermediate language representations were indeed universal and not language specific and could easily be applied to other languages as well. Our approach of combining convolutional neural networks with bidirectional long short term memory cells showed a consistent improvement over baseline CNNs. In general we were able boost accuracy by at least 1%. Furthermore, we established that an Inception-v3 style CRNN outperformed all other approaches both in terms of accuracy (0.96) as well as with noise robustness.

We presented several augmented datasets to evaluate both noise and background music robustness. We were content to note that the noise resistance of the Inception-v3 CRNN reached acceptable levels without us having to modify or restructure our algorithm. This reaffirms our believe in deep learning techniques as a viable tool for audio tasks.

Our CRNN approach interpreted every vector entry along the x-axis as a separate time step for the recurrent layer. Hence, all results gathered here are representative of only ten seconds of an audio file. We believe that in a production ready system we could increase the performance by doing a majority voting across multiple segments of a longer audio file.

Anything else here???

8. Conclusion and Future Work

8.1. Conclusion

8.2. Future Work

- more languages - different transfer learning: progressive neural networksx - songs - phoneme data

Appendices

A. Audio manipulation with SoX

```
1
2     sox -V0 input.wav -n remix 1 rate 10k spectrogram -y 129 -X 50 -m -r -o
          spectrogram.png
3
4
5
6     V0 - verbosity level
7     n - apply filter/effect
8     remix - select audio channels
9     rate - limit sampling rate to 10k; caps max frequency at 5kHz according
          to Nyquist-Shannon sampling theorem
10    y - spectogram height
11    X - pixels per second for width
12    m - monochrome output
13    r - disable legend
14    o - output file
```

Listing 1: Generating monochrome spectrograms with SoX

```
1     scale_factor1=0.94;
2     scale_factor2=0.05;
3
4     sox -m \
5         -v $(echo "$(sox input.wav -n stat -v 2>&1) * ${scale_factor1}" |
              bc -l)  input.wav \
6         -v ${scale_factor2} <(sox input.wav -p synth whitenoise) \
7         -b 16 output.wav
```

Listing 2: Adding white noise to an audio file

References

[1] Abadi, Martín ; Agarwal, Ashish ; Barham, Paul ; Brevdo, Eugene ; Chen, Zhifeng ; Citro, Craig ; Corrado, Greg S. ; Davis, Andy ; Dean, Jeffrey ; Devin, Matthieu u. a.: Tensorflow: Large-scale machine learning on heterogeneous distributed systems. In: *arXiv preprint arXiv:1603.04467* (2016)

[2] Chatfield, K. ; Simonyan, K. ; Vedaldi, A. ; Zisserman, A."Âť: Return of the Devil in the Details: Delving Deep into Convolutional Nets. In: *British Machine Vision Conference*, 2014

[3] Chollet, François: *Keras*. `https://github.com/fchollet/keras`, 2015

[4] Clevert, Djork-Arné ; Unterthiner, Thomas ; Hochreiter, Sepp: Fast and accurate deep network learning by exponential linear units (elus). In: *arXiv preprint arXiv:1511.07289* (2015)

[5] Garofolo, John S. ; Lamel, Lori F. ; Fisher, William M. ; Fiscus, Jonathon G. ; Pallett, David S.: DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1. In: *NASA STI/Recon technical report n* 93 (1993)

[6] Geman, Stuart ; Bienenstock, Elie ; Doursat, René: Neural networks and the bias/variance dilemma. In: *Neural computation* 4 (1992), Nr. 1, S. 1–58

[7] Glorot, Xavier ; Bengio, Yoshua: Understanding the difficulty of training deep feedforward neural networks. In: *Aistats* Bd. 9, 2010, S. 249–256

[8] Gonzalez-Dominguez, Javier ; Lopez-Moreno, Ignacio ; Moreno, Pedro J. ; Gonzalez-Rodriguez, Joaquin: Frame-by-frame language identification in short utterances using deep neural networks. In: *Neural Networks* 64 (2015), S. 49–58

[9] Ioffe, Sergey ; Szegedy, Christian: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: *arXiv preprint arXiv:1502.03167* (2015)

[10] Kingma, Diederik ; Ba, Jimmy: Adam: A method for stochastic optimization. In: *arXiv preprint arXiv:1412.6980* (2014)

[11] Maaten, Laurens van d. ; Hinton, Geoffrey: Visualizing data using t-SNE. In: *Journal of Machine Learning Research* 9 (2008), Nr. Nov, S. 2579–2605

[12] Mishkin, Dmytro ; Sergievskiy, Nikolay ; Matas, Jiri: Systematic evaluation of CNN advances on the ImageNet. In: *arXiv preprint arXiv:1606.02228* (2016)

[13] Montavon, Gregoire: Deep learning for spoken language identification. In: *NIPS Workshop on deep learning for speech recognition and related applications*, 2009, S. 1–4

[14] Pedregosa, F. ; Varoquaux, G. ; Gramfort, A. ; Michel, V. ; Thirion, B. ; Grisel, O. ; Blondel, M. ; Prettenhofer, P. ; Weiss, R. ; Dubourg, V. ; Vanderplas, J. ; Passos, A. ; Cournapeau, D. ; Brucher, M. ; Perrot, M. ; Duchesnay, E.: Scikit-learn: Machine Learning in Python. In: *Journal of Machine Learning Research* 12 (2011), S. 2825–2830

[15] Plchot, Oldrich ; Matejka, Pavel ; Fér, Radek ; Glembek, Ondrej ; Novotnỳ, Ondrej ; Pešán, Jan ; Veselỳ, Karel ; Ondel, Lucas ; Karafiát, Martin ; Grézl, František u. a.: Bat system description for nist lre 2015. In: *Proc. Odyssey*, 2016

[16] Richardson, Fred ; Reynolds, Douglas ; Dehak, Najim: Deep neural network approaches to speaker and language recognition. In: *IEEE Signal Processing Letters* 22 (2015), Nr. 10, S. 1671–1675

[17] Richardson, Fred ; Reynolds, Douglas ; Dehak, Najim: A unified deep neural network for speaker and language recognition. In: *arXiv preprint arXiv:1504.00923* (2015)

[18] Russakovsky, Olga ; Deng, Jia ; Su, Hao ; Krause, Jonathan ; Satheesh, Sanjeev ; Ma, Sean ; Huang, Zhiheng ; Karpathy, Andrej ; Khosla, Aditya ; Bernstein, Michael ; Berg, Alexander C. ; Fei-Fei, Li: ImageNet Large Scale Visual Recognition Challenge. In: *International Journal of Computer Vision (IJCV)* 115 (2015), Nr. 3, S. 211–252. http://dx.doi.org/10.1007/s11263-015-0816-y. – DOI 10.1007/s11263–015–0816–y

[19] Shi, Baoguang ; Bai, Xiang ; Yao, Cong: An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2016)

[20] Simonyan, Karen ; Zisserman, Andrew: Very deep convolutional networks for large-scale image recognition. In: *arXiv preprint arXiv:1409.1556* (2014)

[21] Sizov, Aleksandr ; Lee, Kong A. ; Kinnunen, Tomi: Discriminating Languages in a Probabilistic Latent Subspace.

[22] Szegedy, Christian ; Vanhoucke, Vincent ; Ioffe, Sergey ; Shlens, Jon ; Wojna, Zbigniew: Rethinking the inception architecture for computer vision. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, S. 2818–2826

[23] Wermeister, Thomas ; Herold, Tom: *Practical Applications of Multimedia Retrieval: Language Identification in Audio Files.* `https://github.com/twerkmeister/iLID/blob/master/Deep%20Audio%20Paper%20Thomas%20Werkmeister%2C%20Tom%20Herold.pdf`, 2016

[24] Zazo, Ruben ; Lozano-Diez, Alicia ; Gonzalez-Rodriguez, Joaquin: Evaluation of an LSTM-RNN System in Different NIST Language Recognition Frameworks. In: *Odyssey 2016* (2016), S. 231–236

[25] Zhang, Yu ; Chuangsuwanich, Ekapol ; Glass, James R.: Extracting deep neural network bottleneck features using low-rank matrix factorization. In: *ICASSP*, 2014, S. 185–189